KTH Royal Institute of Technology

MF2058 Mechatronics, Advanced Course

The Juggling Robot Project - JuRP

Yared Efrem Afework, Dennis Lioubartsev,
Ahmed Moustafa, Ludwig Nyberg, Nilas Osterman,
Aditya Pal, Fredrik Schalling, Max Thiel

supervised by

Pouya Mahdavipour

Abstract

Toss juggling, a skill long since mastered by humans, presents an interesting challenge in the field of robotics. Designing, constructing and implementing a robotic system capable of object manipulation requires the successful integration of a number of fields, including but not limited to mechanical and electronic design, computer vision, control theory and kinematics. This report details the work done by the Juggling Robot Project HK (JuRP-HK) team in their efforts to build a robot capable of throwing and catching a single ball using a single arm. The team based their design choices on a previous State-of-the-Art analysis, and in accordance with a set of requirements set by the stakeholder.

The arm built was a serial mechanism arm with five degrees of freedom, including a two degrees of freedom cup-stabilizing system as the end-effector and a three degrees of freedom robotic arm. The actuators were locally controlled using manually tuned Proportional-Integral-Derivative controllers, working under the command of a high level controller. The high level controller uses ball positional data collected by a proprietary motion capture system. The positions captured are used to estimate the ball trajectory of the ball, and the inverse-kinematics joint angles are using the Levenberg-Marquardt algorithm. A Linear Quadratic Regulator is used to calculate the optimal path joint-wise necessary to get the robot's end-effector into position to catch the ball.

In the end the robot was capable of performing throwing and catching of a ball. Doing both in one movement was also shown to be possible, though with a lower rate of success.

Acknowledgements

We would like to thank the entirety of the KTH JuRP research group for presenting this task, with a special thanks to our supervisor Pouya Mahdavipour Vahdati for invaluable feedback and guidance.

Furthermore, we would like to thank Staffan Qvarnström and Thomas Östberg for all the support and help given withm the electronics and hardware.

Finally we would also like to express our gratitude to Augustin Crampette, for his work in creating Simscape simulations that we based our own simulations of off, and Sjoerd Koopmans for setting up the Motion Capture system as well as the ROS middleware to interface with it.

Team JuRP-HK
December 2018, Stockholm

Contents

Nomenclature

DOF Degree(s) of Freedom

DOF Degrees of Freedom

EE End Effector

HIL Hardware in the Loop

HK [Mechatronics] Higher Course

HLC High Level Controller

IK Inverse Kinematics

IMU Inertial Measurement Unit

IR Infrared

$JuRP$ Juggling Robot Project

KTH the Royal Institute of Technology

LCM Lightweight Communications and Marshalling

LED Light-emitting Diode

LLC Low Level Controller

LQR Linear Quadratic Regulator

$MoCap$ Motion Capture

MPC Model Predictive Control

PCB Printed Circuit Board

PID Proportional Integrating Differential controller

PWM Pulse-Width Modulation

ROS Robot Operating System

TCP Transmission Control Protocol

$UART$ Universal Asynchronous Receiver-Transmitter

UDP User Diagram Protocol

1 Introduction

1.1 Background

The art of juggling - a highlight of many a circus act, the showcasing of which can awe an entire crowd to silence. Beyond its pure entertainment value it presents a complicated engineering task for robticists to replicate, engaging multiple technical fields in not only mechanics and electronics, but even more so perception, planning and control.

The diversity of topics and problems that one is forced to evaluate in the process of synthesizing a human juggling motion makes it an academically interesting and rewarding project to do research in, as the gained knowledge can be useful in a broad range of other applications.

The original Juggling Robot Project (JuRP) is a project at KTH Department of Machine Design led by Professor Martin Törngren. The team decided to present this problem to the Mechatronics capstone course, resulting in this project group: *JuRP-HK*.

1.2 Project Description

The goal of this project is to develop a serial mechanism robot that can juggle balls using a closed loop vision system. The necessary spatial information to perform the juggling is to be extracted using a Motion Capture system called Vicon Tracker^{TM}by Vicon Motions System.

As mentioned previously, this project is a down-scaled and delimited version of the larger task approached the JuRP team, who have tasked this group with creating a prototype which can serve as a test environment for the results of their own research. Thus, this project is owned by KTH and the JuRP team serve as the stakeholders.

1.3 Requirements and Delimitations

A series of high level requirements were presented to the group at the start of the project. They have been interpreted and translated into a series of technical requirements as follows:

1. The End Effector (EE) shall have a reach of a half circle in front of it with a radius of 80 cm in the catch-plane.
2. The EE of the robot shall be able to move with a speed of 7 m/s up and down along the vertical Z-axis.

3. The EE of the robot shall be able to move 5 m/s in the horizontal X-axis and Y-axis respectively.
4. The robot shall be able to toss a ball 2.5 meters above its base.
5. The EE of the robot shall be able to decelerate vertically from 7 m/s to 0 m/s in 0.5 s (14 m/s^2).
6. The EE shall be able to tilt 30 degrees in any direction without dropping the ball.

The following is a list of delimitations provided to the group by the stakeholders.

- Only electromechanical actuators shall be used.
- In-hand manipulation of the balls is not required, i.e. it is sufficient that each hand can hold only one ball at a time.
- The prototype will have a fixed base.
- The balls used will be regular (non-bouncing) juggling balls.

Finally, a budget of 25 000 SEK has been provided to the project.

1.4 Report Organization

The report is organized as follows.

Section 1 Introduction introduces the reader to the subject and background of this report. The short overview is followed up by a description of high level requirements which defines the outline of the engineering task, these are then further defined into testable technical requirements. Since the there are a large variety of solutions to the problem some delimitations are also set to limit the scope of the project.

Section 2 State of the Art contains a theoretical background of the relevant technical areas. This literature review is carried out to give a rough outline of how to structure the process and design by analyzing comparable designs. The outline is later on used to choose a set of solutions to be evaluated for the final product implementation.

Section 3 Methodology discusses how the project is managed. The project is divided into different architectural subsystems to simplify further task management. Also, a few set of project management methodologies are evaluated and then implemented based on a set of criteria of what ought to fit the team and development scope best.

Section 4 Design Choices details the design choices made and why they were made. Following an investigation and compilation of the current State of the Art, it was time to compare various solutions in the design space and make decisions regarding what would actually be implemented in this project's robot.

Section 5 Implementation presents the implementation specifics of the various subsystems, including the specific mechanical design, the electronical architecture, the software middleware communications, and the different level controllers.

Section 6 Verification and Validation examines how the total system can be verified and validated against the technical requirements and stakeholder requirements set by the group itself and its stakeholders respectively.

Section 7 Results provides objective performance results for the robot, as well as verification and validation verdicts.

Section 8 Discussion and Conclusion wraps the project up with a discussion regarding how successful the engineering problem is solved in different architectural levels. Questions regarding performance are analyzed and answers are given based on the result section. The discussion is finally concluded with high level thoughts of the entire project.

Section 9 Future Work finally presents potential avenues of future work for the project, such as possible solutions that were not implemented in the scope of this project but which the group believes would serve to improve it.

2 State of the Art

A literature study was done at the beginning of this project in order to a get an understanding of what is currently the State of the Art in the field of juggling robots.

2.1 Mechanical Design

Robotic arms are generally built following two different mechanical principles, namely serial and parallel mechanisms. When an arm is composed of a combination of both, it is called a hybrid mechanism. A detailed comparison of the different approaches are given in [1].

2.1.1 Serial Mechanisms

A serial mechanism consists of a chain of kinematic links from its base to an EE, where each joint is connected to another joint serially. The number of DOF is equal to the number of actuated joints. A robotic serial mechanism arm is commonly designed in a human-like fashion, and an example of this is the Selective Compliance Articulated Robot Arm (SCARA) shown in Figure 1.

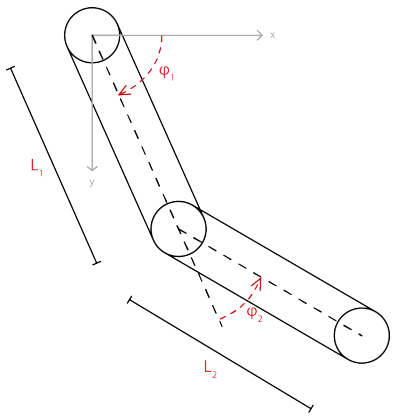

Figure 1: A visual representation of a SCARA mechanical manipulator.

Serial mechanisms have several advantages over parallel mechanism, including:

- ease of solving the forward kinematics problem [2],
- a reasonable complexity of the inverse kinematics problem,

- a larger work-space.

Additionally serial mechanism offer a module based approach since additional actuators can be added when needed thereby increasing the DOF for the arm.

One issue with serial mechanism, especially the SCARA design, is that they rely on each respective joint actuator to carry the cumulative load of the actuators connected after it, all the way to the final joint carrying the EE. This may lead to a loss of accuracy and rigidity due to errors accumulating serially at each joint from the base to the EE [3]. It is therefore of utmost importance to design the arm to be rigid, in order to be able to maintain the controllability and precision of the complete system.

2.1.2 Parallel manipulators

Parallel manipulators consists of a number of links connected in parallel via different types of joints to control a single platform [4], where the number of DOF is determined by the amount of links connected to the EE. Every link is driven by one actuator and is usually placed on, or nearby the base of the manipulator. Depending on the desired function of the mechanism, different types of joints are used to enable movement or rotation in the desired directions.

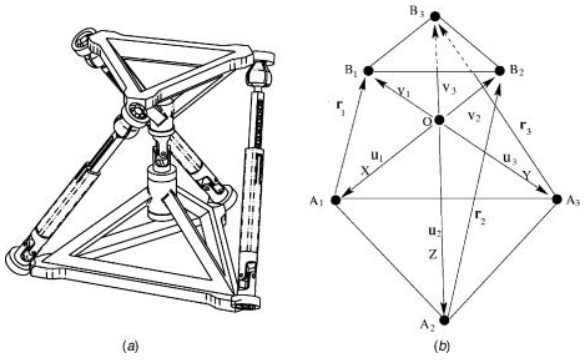

Figure 2: A 3-DOF parallel manipulator and the coordinate system of the allowed movement [5].

Figure 2 shows a model of a 3-DOF parallel manipulator with a platform base on the bottom and the EE on the top. The limbs are connected to the base and actuates to move the EE according to the coordinate scheme also presented in the figure.

The benefits of parallel mechanisms over serial mechanisms ([6], [7], [4]) include:

- they allow for a greater load carrying capacity due to having multiple arms to distribute the weights over,
- their low inertia, making them easier to control;
- a built in redundancy,
- higher precise due to not suffering from the same error accumulation that serial mechanisms do.

Parallel manipulators are also referred to as closed-loop system due to their inherently inter-connected mechanical structure [6]. This makes the kinematic analysis and modelling highly complicated due to the system of closed loop differential equations. The solution to the inverse kinematic problem can often end up in singularities, making control and path planning difficult. Other downsides are the limited work-space due to the constraints caused by the mechanical structure [4], as well as the non-modular approach.

2.1.3 Hybrid Mechanisms

The previously mentioned serial and parallel mechanisms can be combined into a hybrid system. A hybrid system is used when there is a desire to combine the advantages of the serial and parallel mechanisms. When designed properly it can result in greater work space volumes, higher stiffness, greater precision and better load-bearing capabilities [8].

The highly complex kinematics of the parallel manipulator remains as a disadvantage for hybrid mechanisms, inciting difficulties with the inverse kinematic problem and control [6].

2.2 Control Design

One of the corner stones of a highly precise and fast robot arm is an effective control algorithm. There are various approaches to achieving a satisfactory control design, some of the more relevant are discussed in this section.

The standard control problem can be formulated as: given a plant (G) with a controller (F) create a control signal (u) such that the output (y) follows the reference signal (r), while taking into account disturbances (w) and measurement noise (n). Figure 3 visualizes the control scheme as a flow chart.

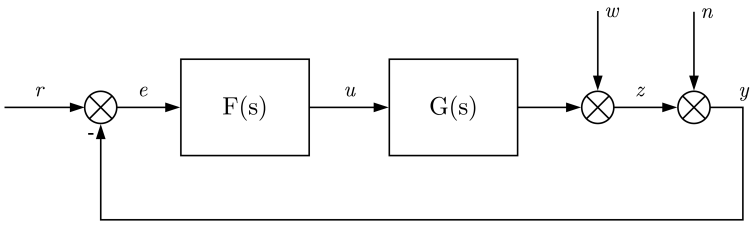

Figure 3: Plant (G) with a controller (F), the control error (e) is the difference between input and output, $e = y - r$.

In many cases it is also desirable to have the control signal as small as possible. The reason being that often the usage of the control signal comes with a cost, for a motor it can be fuel consumption or battery depletion. This results in an optimization problem as follows,

$$\min(||e||^2 Q_1 + ||u||^2 Q_2)$$

Where Q_1 and Q_2 are weights that are set depending on the desired compromise between minimizing the control error and minimizing the control signal. This is just one example of a cost function; it can be expanded as necessary to optimize more parameters.

There are several ways of regulating such an optimization problem, including but not limited to the Linear Quadratic Regulator (LQR), the Linear Quadratic Gaussian Controller (LQG) [9] and the Model Predictive Controller (MPC) [10]. Simpler methods have also been proven to work in [11], [12] where trajectory planning has been combined with Proportional-Derivative (PD) or Proportional-Integral-Derivative (PID) controllers. This however reduces the task from an optimal control problem to a classical control approach.

Combinations of the above solutions can also be implemented, such as in [13] where an MPC has been implemented as a High Level Controller (HLC) with a PID controller as a Low Level Controller (LLC) in order to regulate a nonlinear reactor-separator process. Another example is [14] where the control structure of a Vertical Take-Off and Landing aircraft is designed with a combined HLC and LLC structure. In both of these examples the HLC is responsible for the overall situational awareness and mode selection while the LLC, closer to the hardware, controls the actuators.

In the case of the juggling robot the HLC would be responsible for the motion planing of the EE based on feedback from the vision system, while the LLC would be responsible for the actuator movement of the robot's various joints.

2.3 Software and Information stream

When designing a time critical system effort needs to be put into ensuring that the data is gathered and the information distributed to other subsystems in a fast and reliable manner, in order to ensure the responsiveness of the system. The proprietary system from Vicon and methods for setting up the distribution of information hence need thorough analysis.

2.3.1 Vicon Motion Capture Systems

In order to continuously keep track of the positioning of the ball and the robot in space a Motion Capture (**MoCap**) camera system is used. It provides visual feedback to the main computational unit and robot that is used to control the position of the EE.

The MoCap system, composed of six cameras is specifically built around the software know as Vicon Tracker [15]. The cameras emit a ring of infrared (**IR**) light using light-emitting diodes (**LED**). The cameras are sensitive to IR light, and by clothing a juggling ball in a special reflective fabric which will reflect back the emitted IR light, the ball can be tracked by the MoCap system. The positional information is sent via a Transmission Control Protocol (TCP) network to the user PC. On the user PC, virtual representations of each tracked physical object (e.g., "a juggling ball") in the Vicon work-space [16] will have their positional information continuously updated.

2.3.2 Communication

When sending data from one process to another there has to be a mutual agreement on the appearance of the data being sent. This is regulated through a protocol which regulates the structure and the content of the sent data packages, combined with a method for handling errors. Common protocols for data transport include two categories: connectionless communication with emphasis on reduced latency over reliability, and host-to-host connection oriented communication where a receiver of corrupt data can demand a re-transmission. User Datagram Protocol (UDP) is an example of the former category, and TCP an example of the latter. Building communication based directly by above protocol is efficient and robust for high demanding applications as proved in [12].

In distributed applications, one solution is to add an abstraction layer between the transport layer and the application layer. This layer, called middleware, is a software that enables communication and the management of

data between systems. The software is responsible for keeping up a seamless infrastructure between information sharing systems, i.e. the *-to-* in peer-to-peer. By using a framework in setting up the communication infrastructure one is able to work modularly with systems without having to reconfigure peers when the system or parts of it are replaced. The middleware often emulates a communication pattern to the user to simplify visualization and understanding of system behavior. The system side configuration is often done with an actual peer-to-peer communication with a common protocol such as UDP or TCP [17].

The most commonly used structure for middlewares in interprocess communication is the *Publish-subscribe pattern* [18], [19]. In theory the senders (publishers) are not configured to send to specific receivers (subscribers), the communication is instead set up by using an intermediate entity (topic) whereby senders publish messages without any knowledge about which processes might be subscribing to its information on the other side, as seen in Figure 4. This structure centralizes the configuration and administration of communication and is often implemented by using a middleware as seen in Figure 4. Two commonly used implementations of this pattern include the Robot Operating System (ROS) [20] and Lightweight Communications and Marshalling (LCM) [21].

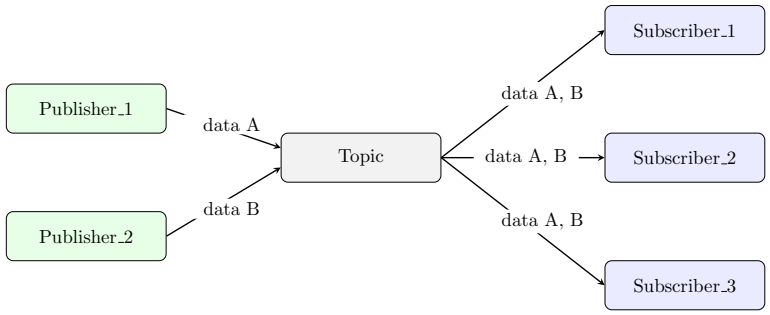

Figure 4: Publisher-Subscriber pattern structure layout.

3 Methodology

The project can conceptually divided into five different tasks (excluding the writing of this report):

1. ***The Arm*** The mechanical design and construction of the arm and overall frame.
2. ***Actuation*** Choice of motors, power supply, sensors and microcontrollers to handle the low-level control.
3. ***Estimation of catching position*** Involves the recording of the spatial information from the camera system, the estimation of an approximate ball trajectory and the outputting of a catching position and time for the EE using inverse kinematics.
4. ***Arm movement trajectory*** Producing a trajectory to move robot arm's individual joints optimally in order to get the EE into position. The challenge here is implementing an algorithm which can not only be constrained to avoid producing certain joint paths and positions that might jeopardize the safety of the robot, but which can also do so without violating the time constraints.
5. ***Communications*** Organizing and verifying all communications between the various systems.

Each of the five areas were separately broken down into concrete sub-tasks that could be assigned to a person.

3.1 Project Management

To organize the project two commonly used methodologies were discussed, whether to work agile or whether to adopt a waterfall approach.

3.1.1 Agile Approach

One realization of working agile is the framework Scrum which places focus on constantly delivering end user value by iteratively adding functionality. By working iteratively a continuous evaluation of the process is enabled. Through this approach, the project is forced to have distinct deadlines for a working solution that ought to be evaluated.

Furthermore, the core of working with Scrum is the "Daily scrum" that follows a defined agenda. Every team member presents what they have done since the last meeting, plans to do until next and any potential issues that could prevent one from accomplishing what is planned

3.1.2 Waterfall Approach

The waterfall approach centers around a sequential process starting with defining product requirements followed by design, implementation and finally verification. The methods origins from handling the dilemma of exponentially increased costs of design changes as a project reaches deadline. The waterfall approach also enables scheduling checkpoints to test and make sure requirements are fulfilled through unit to system tests. The process of continued testing of is often entitled "the V-model".

Typical issues by working in strictly this manner is the need of defining end requirements at the beginning of the project; many end users changes as the project progresses. Changing requirements could potentially result in huge costs if the current design do not support the new need.

3.1.3 Implemented Management Strategy

When discussing the various approaches, the consensus was that an approach that would facilitate working to meet external deadlines, fulfilling the given high level requirements and clear distinction between areas to enable easy sharing of work. The management should also be easily understood by all members and quickly implemented. As the requested deliverables given to the team were very clear from the start, and the stakeholder has deep knowledge in the subject, it was concluded that those requirements will not change throughout the project.

Thus, the choice fell on using a high level waterfall approach to the engineering problem. The project was divided into several sequential phases seen below and each broken down and put in a time-line seen in Appendix B.2.

1. Evaluate different design alternatives
2. Simulate system to tweak and finalize design
3. Implementation
4. Verify and Validate

To make it possible to work in parallel and avoid bottle necks, work was started on a previously made 1-DOF model provided by the stakeholder, allowing part of the group to work iteratively on the low level control and actuation software and on closing the loop trhough the camera system before the mechanical design of the final protype had been set up. This enabled the team to have functional software units when the multi-DOF mechanical arm was assembled.

The team agreed to adapt parts of the Scrum format by meeting formally twice a week for quick progress update. The meetings consisted of each person presenting their current state on the format

1. Achieved progress
2. Plan forward
3. Faced problems

By this model the team kept up progress and had a platform or channel to gain insight in areas other the one currently engaged in. Also, this enabled the team to distribute work dynamically and one to move between different architectural areas without friction.

Finally, at the time for evaluation of the set requirements compared to the achieved product, a follow up by an analysis and summarize of potential future work was conducted.

4 Design Choice

Based on the State of the Art analysis a collection of possible solutions were compared to find the most suitable design choice. These findings are compared in the following section in order to motivate the final choice of design.

4.1 Mechanical Design Adoption

The result of the literature search presented in Section 2 is summarized and presented in the table below.

	Serial	Hybrid	Parallel
Speed & Force	High	High	High
Work space	Very Big	Big	Varies/Limited
Robustness	Less Robust	Varies/Robust	Very Robust

Table 1: Main mechanical points considered when making the decision on the mechanical structure of the robot arm.

The conclusion from the literature search was that a serial mechanism is the most suitable kinematic mechanism type for this project. The larger work-space effectively increases the capability of the juggling robot compared to a parallel design. It can be designed to be very fast depending on the actuators chosen for the different joints.

The serial mechanism is however not as rigid as the other designs due to the load placed on each joint, which could lead to inaccuracy in the position of the EE. A serial mechanism of this scale will be rigid enough for the specified task. In addition, any problems can be compensated for by utilizing the accuracy of the Vicon camera system during feedback and designing the EE to be able to counteract positional errors by increasing the area of the hand, making it able to catch the ball even though the EE position is not precisely at the ball's landing point.

The serial mechanism will provide a human-like appearance for the robot and will also enable a modular approach to the construction, making the robot easier to assemble and design as well as pleasing the stakeholders.

One of the biggest benefits of using a serial mechanism is the lower complexity when calculating the inverse kinematics of the EE position. This will lead to less computational effort for the controller and trajectory planner.

Finally, the literature study revealed that serial mechanism manipulators have previously been chosen to perform similar tasks which greatly helps

sway the decision in its favour.

4.1.1 End-Effector Design

The design choices governing the EE are focused on making a simple contraptions that lacks a gripping mechanisms. This resulted in the group deciding to go for a cup-shaped EE. The two designs chosen for the specified task were a cone or a bowl.

A bowl, while better for catching the juggling balls due to its wider brim-to-mass ratio, suffers from an uncertainty problem: the ball is not as certain to be caught and positioned in one defined position. A cone on the other hand ensures that the ball is guided into one specific place, namely the center of the cup, making the overall behaviour of the catching and throwing of the ball more deterministic.

By making the surface of the cone relatively smooth, the ball will always roll down to be centered when placed in the EE, making the cup the superior choice of design.

4.2 Control Adoption

The choice was made to utilize an HLC in combination with an LLC to control the robotic arm. This was due to the fact that no fully representative model of the system is able to be created due to scarce information regarding motors and the iterative design process for the mechanical design. Hence, by utilizing a combination of an HLC and an LLC the system is able to be fully controlled without an accurate model of the system dynamics.

Table 2 below shows a summary of the various control strategies presented in Section 2.2.

	PID	MPC	LQR
CPU Demand	Low	Very High	High
Adaptive/Static	Static	Adaptive	Adaptive
Complexity	Low	High	Medium

Table 2: A comparison of three different control algorithms.

The decision has been made to use LQR controller as an HLC, and a PID controller as an LLC. The reason for discarding the MPC as an HLC is that the MPC is computationally more expensive [22] which can lead to difficulties and latency issues between the different components in the system due to the computation time required at each sampling period. On the

other hand, the LQR controller is sufficient for the control problem for the task of the juggling robot without being as computationally expensive. The Arduino uno will be implementing one PID controller per actuator, while the PC will be dealing with the HLC.

If non-satisfactory results are achieved using the LQR, the controller can be extended to a MPC design utilizing the already designed LQR controller with a more detailed model of the system dynamics. This approach result in a methodical approach to the control design, with possibility of extension for future work.

4.3 Communication

The criteria used to compare and pick the communication protocol is presented in the following list, and the actual comparison in Table 3.

- *Modularity*
  Does the method support a change of components without any additional configuration?
- *Performance*
  Does the method imply additional processes running compared to using another method, i.e will it require additional computation and/or run time?
- *Community*
  Subjectively decided based on recommendation from supervisors and staff in the faculty.

Modularity is evaluated based on whether or not it is possible to switch components without any additional configuration.

	ROS	LCM	No middleware
Modularity	High	High	None
Performance	High	High	Very High
Community	Average	Low	Very Large

Table 3: Overview of evaluated methods for implementing communication

ROS was picked due to fulfilling the modularity criteria as well as the fact that it has a larger community than LCM.

5 Implementation

The overall system overview can be seen in Figure 5. It starts with the Vicon camera system that tracks the ball in the world frame. It is also used to verify that the EE has reached the desired position. Data from the Vicon system is then passed on via the Vicon PC down to the Ubuntu computer, it is on this machine that the supervisory controller is running. In the lower left subsystem the low level controller can be seen, this is the controller that is responsible for performing the movement generated by the supervisory controller. The last subsystem is the physical arm; as can be seen the EE is its own closed loop since it is stabilized utilizing an Inertial Measurement Unit (IMU) and servo motors. Communication between different nodes and different subsystems located on different machines is done mainly using ROS and serial Universal Asynchronous Receiver-Transmitter (UART) communication. Systems, subsystems and communication are all discussed further in this section under subsections 5.1 - 5.4.

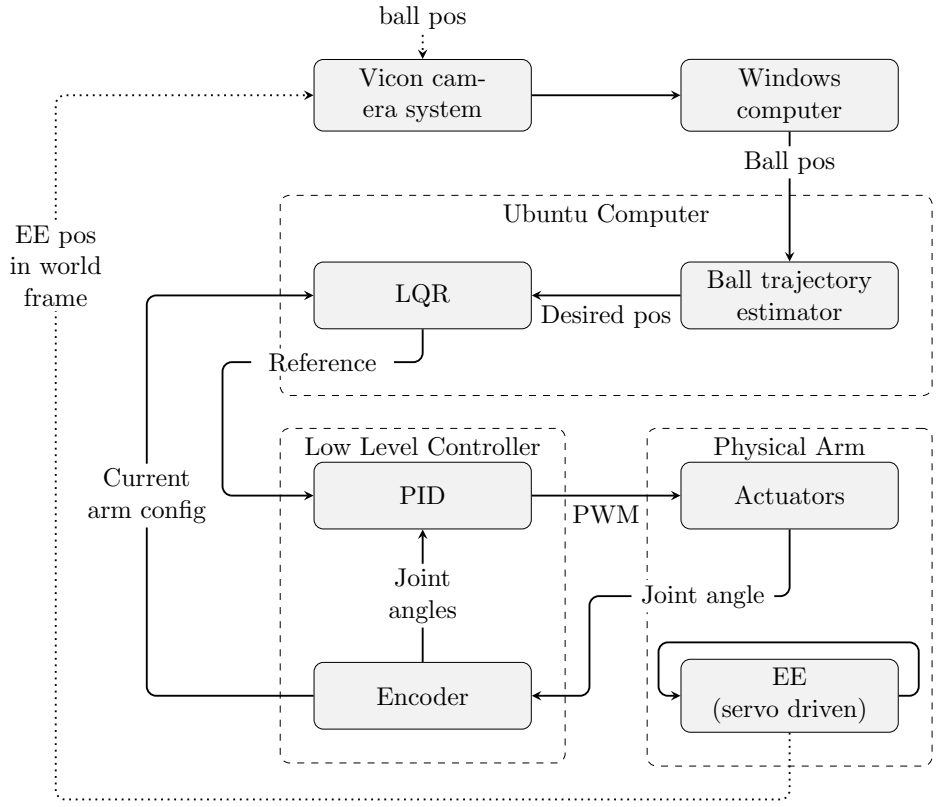

Figure 5: System overview, subsystems are grouped within dashed lines and the dotted line represents the data that the Vicon system generates from sampling the ball position and EE position.

5.1 The Mechanical Arm

The initial requirements of the project were to develop a juggling mechanism to perform basic level juggling (i.e., throw and catch a ball). There were several processes involved in designing the mechanical structure for the arm. This was initiated by creating a basic serial mechanism and distributing the degrees of freedom at the appropriate locations. The main challenge following that was selecting the motors for implementing the degrees of freedom in the serial mechanism. This was done using simulations made in Simulink and MATLAB. After selecting the motors, CAD models were created which helped to develop other parts like motor mountings and the links to connect them. CAD models were drafted and tested for excessive stress concentrations. Materials were also selected accordingly.

5.1.1 The Serial Mechanism

The initial requirements regarding the robotic arm demanded an extensive analysis of the different types of mechanisms commonly used in this field. Specific cases of juggling robots were used to understand the limitations and the advantages of the already existing systems.

The basic serial mechanism of a robotic arm has 2 degrees of freedom. In order to utilize the maximum possible work-space and not complicate the structure beyond realistic limits, a 5-DOF serial mechanism was decided upon. As shown in Figure 6, the 5 different actuators would help the EE to move in 3-D space. This formed the basis for the Simulink model and eventually, the complete CAD model to fulfill the functional requirements of the arm.

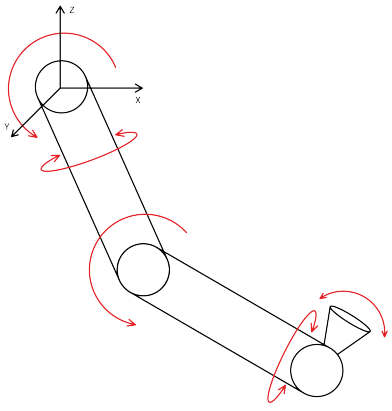

Figure 6: The basic serial mechanism map with 5 Degrees of Motion.

5.1.2 Simulations

The physical arm is simulated in MATLAB and Simulink. A simulation is made to quickly iterate the arm design until the requirements presented in Section 1.3 are satisfied. MATLAB and Simulink are chosen because they offer a model-based approach when modeling the arm.

The team got help from Augustin Crampette who helped setting up the forward- and inverse kinematic models of a five link serial arm in Simulink. The Simscape toolkit is used in Simulink to simulate the dynamics of the system. The length of the arm is set to satisfy requirement 1. To verify the model, three separate reference inputs are evaluated. The references are the worst case linear movements in x, y and z axis, corresponding requirements 2 and 3. The set-up is shown if Appendix C.

The requirement 2 of moving the EE with a speed of 7 m/s along the Z-axis is the most critical requirement for the actuators. The actuators in a serial mechanism move only the the components mounted further out on the mechanism. This means that the actuators close to the EE should be smaller than the actuators close to the base. Therefore it is natural to start with the EE and move inwards to the shoulder. The weight of the EE was estimated based on the weight of a ball with some extra weight for the ball holder. The corresponding weight is added to the EE in the model. The model is run for the reference movements showing the required torque and speed for each actuator. Actuators fulfilling the torque and speed are found and the approximated weight of an satisfying actuator is added to the model. Some extra weight is added to gain error margin. This method is iterated to get the approximated weights of all actuators and links.

5.1.3 Arm Design (CAD)

Solid Edge was used to continuously design the parts, in sequence, starting from the elbow motor. The biggest challenge was to design strong mounts for the actuators that could withstand the peak vibrations and torque during the throwing motion. The two major loads were the weight of the actuators and the impact of the inertia of the joints during peak accelerations. Initially, the upper arm and the forearm were drafted by sequentially adding parts to the elbow motor.

Each of the mounting parts for the motors were analyzed for stresses using Solid Edge Simulation. Steel and aluminum were the two materials that were analyzed. A trade-off between the weight of the part and the strength had to be made, especially for the mountings for holding the motors in place.

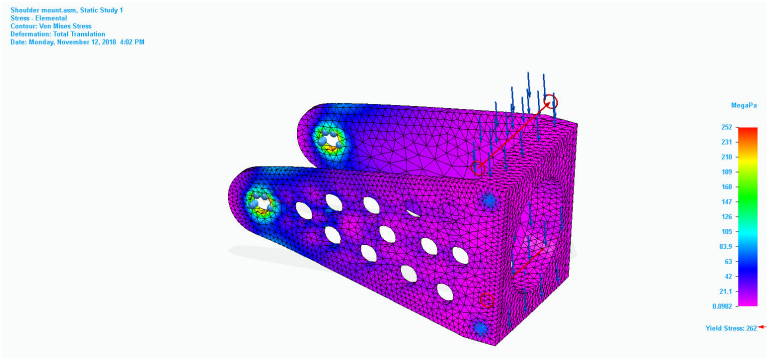

Figure 7: Stress analyzes on the part to be mounted on the shaft of the Shoulder Motor.

5.1.4 Tube Design (CAD)

There was a minor dilemma during the selection of a proper link between the Elbow and the EE. Carbon fiber tubes and carbon fiber rods were analyzed in Solid Edge to check for stresses on static loading at one end and fixed at the other. The link was required to be lightweight and strong enough to support the EE and the ball, especially during impact. The designs were analyzed for stresses in Solid Edge. The tube design was a better option and was therefore selected, allowing the further connecting parts to be designed.

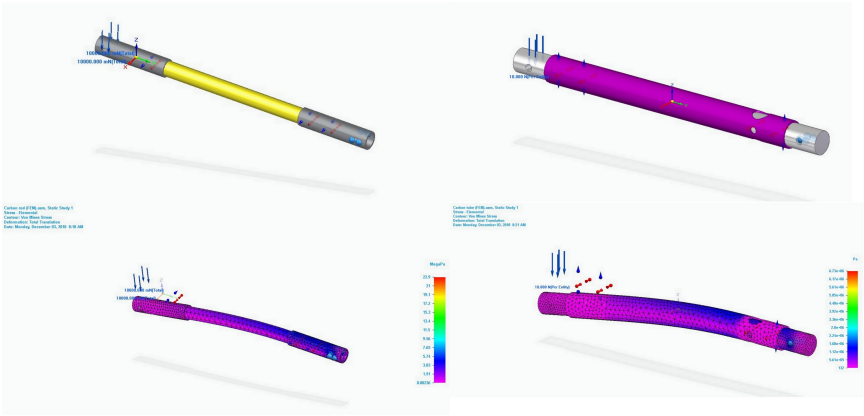

Figure 8: Stresses on the carbon fiber rod (left) compared against the carbon fiber tube (right).

Apart from the structure, the material was also analyzed. Aluminum was the alternate material that was thought upon. But, in order to minimize the weight of the links between the motors, carbon fiber was chosen.

5.2 Actuation

One of the original requirements given from the very start of this project was that only electromechanical actuators be used, thereby excluding hydraulic and pneumatic actuators. Furthermore, as a completely serial mechanism arm was chosen only rotational motors were used.

A system overview of the actuator setup is given in Figure 9. This setup is the same for all three actuators of the arm.

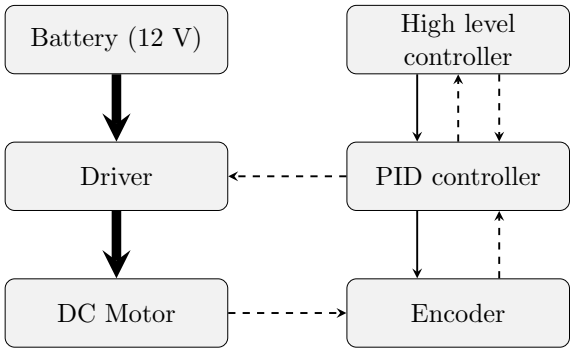

Figure 9: A graphic illustrating the flow of power and information in the actuator. The dashed lines represents information, the thick lines 12 V and the thin lines 5 V.

5.2.1 Motors

The choice of motors was based on simulations made in MATLAB and Simulink. In order to perform a fast enough motion of each joint the motors would have to produce sufficient torque and speed. Since the actuator located in the elbow is responsible for the tossing motion, most focus was put on finding a suitable actuator for that joint. The group compared different electromechanical motor types, such as servos, stepper motors and DC motors before finding a suitable fit for the elbow motor - the DOGA 319.9059 DC motor which has a very high starting torque and suitable speed ratings for this application. The motor does not have any built-in encoder which meant that external sensing would have to be used. The two remaining motors were provided by the department and the datasheets were read in order to verify that the speed and torque for each actuator were sufficient. Also, the performance of each actuator were tested to further assure the choice of the actuators.

5.2.2 Drivers

Pololu G2 High-Power line of motor drivers were chosen, specifically the 18 V maximum battery voltage and 25 A continues current model, for the following reasons:

- It is very simple, functioning like an H-bridge accepting a PWM-signal for input
- It can handle high currents (25 A continuously, 60 A peak), providing a margin of safety in case the appropriated Shoulder and Upper-Arm Motors ended up being weak and needed to be compensated with high current supply
- It is compact and light-weight, allowing for convenient mounting on the actuators
- It provides a current sensing output pin, allowing for potential feedback control using the current.

The driver accepts a PWM signal with a duty cycle between 0% and 100%, as well as a digital direction pin. Setting the direction pin HIGH or LOW will result in clockwise and counter-clockwise movements. The driver is supplied power from batteries, such that a 50% duty cycle will supply the motor with a voltage level of 50% the rated battery voltage.

5.2.3 Encoders

A magnetic encoder was used for the external sensing of the actuator, specifically an RLB rotary miniature PCB level incremental magnetic encoder from RLS Renishaw. It comes in a two-component package, a reader head and magnet ring.

The magnet ring is attached on the moving shaft of the actuator, and the reader head an appropriate distance away from it. The reader head connects to an Arduino Nano, which powers it and reads the A and B channels. There are 2304 counts from the the encoder for every full revolution of the actuator shaft. In addition to 5 V, GND, A and B the reader head provides an Error channel and a Z channel (also known as zero, index or reference channel) but these are not actively used. By resetting the microcontroller, the encoder count is reset and a new zero angle is set at the current position.

The Arduino Nano connects to the controller, receiving power from it and sending encoder values over the serial UART protocol. The encoder software relies on pin-change interrupts and it was decided upon that keeping it

on a separate microcontroller would ensure that the interrupts would not interrupt the controller software and cause unexpected behavior.

5.2.4 Power Supply

12 V car batteries are used to supply the actuators with power. The advantage of using powerful batteries for the actuators is that they can accommodate a current draw of hundreds of amps. There was an initial doubt that the motors chosen would be performing less than what the simulations had reported as necessary for the final arm design. A measure to counter this risk was to use powerful car batteries that could ensure the motors were not being limited in power electrically, even if it would come at a potential cost of motor lifetime degradation. A reduction in lifetime would be preferable compared to a complete failure to reach the goals of this project.

Other options included using lab bench power supplies, or rectified mains power. The lab bench power supply modules available at KTH have a current output limit of 4 A. Each module has two ports, allowing for 8 A if placed in parallel, and multiples of that if more of them are used. However this was deemed as a poor solution since the required current would require far too many modules.

For the elbow motor, to ensure sufficient maximum speed of the EE in the throw of the ball, two 12 v batteries were placed in series to reach 24 v.

5.2.5 End Effector

To be able to throw the ball in a straight vertical line an EE was built with the purpose to keep the direction of the ball leveled regardless of movement and pose of the arm.

Figure 10 show the overall design of the EE. It is composed of a cone-shaped cup that will be able to catch the ball and place it centered at the axis during the throwing motion. The cup is made using 3D-printers resulting in a slippery surface the ball can glide on causing it to always be centered in the cup.

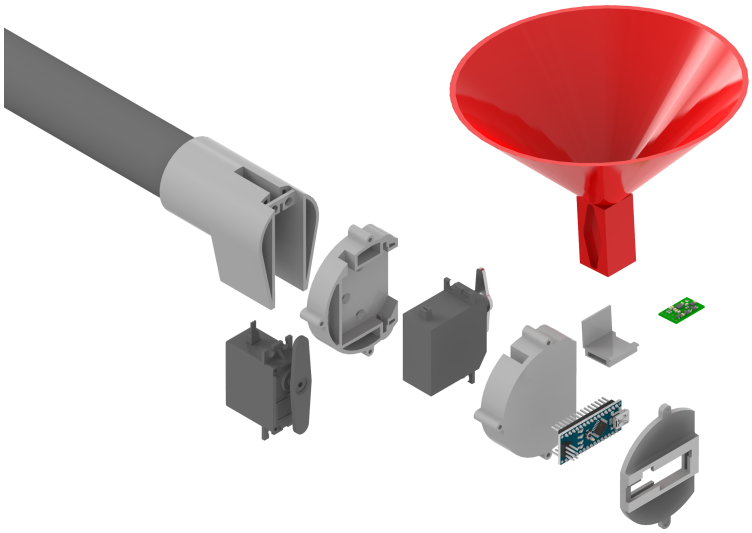

Figure 10: An exploded view of the End-Effector design

The cup is connected to two continuous rotational servos providing a two DOF motion, which are able to control the roll and pitch of the EE. The structure holding the servos is made in 3D-printed PLA plastic. The system utilizes a Arduino Nano as microprocessor which handles the control. An IMU is integrated in the system providing 9-axis accelerometer, gyroscope and compass to calculate the angles which serves as feedback to the microprocessor.

The entire EE is driven using four AA batteries and a small circuit containing a 5 V regulator which supplies power to the servo motors as well as the Arduino Nano and IMU making the system entirely independent.

5.3 Communication

Controlling a multi-platform system requires robust communication. To facilitate this the middleware ROS was used. A ROS network is built up by a core node, *ROS Master* participating systems, *ROS Nodes* and communication entities, *ROS Topics*. A ROS Node in the network can be set up as a publisher and/or subscriber to a ROS Topic and through this data is transmitted between systems as shown in Figure 11.

In ROS there are two possible protocol choices for the peer to peer packets transmission, UDP or TCP. TCP provides an ordered and error checked stream of bytes while in UDP the two way communication with error checking is given up for faster transmission times. Even though UDP generally is

preferred in systems with continued data feed, TCP is mainly used in this project since the communication with external systems is limited to UDP. With this middleware setup, a near real time system is achieved.

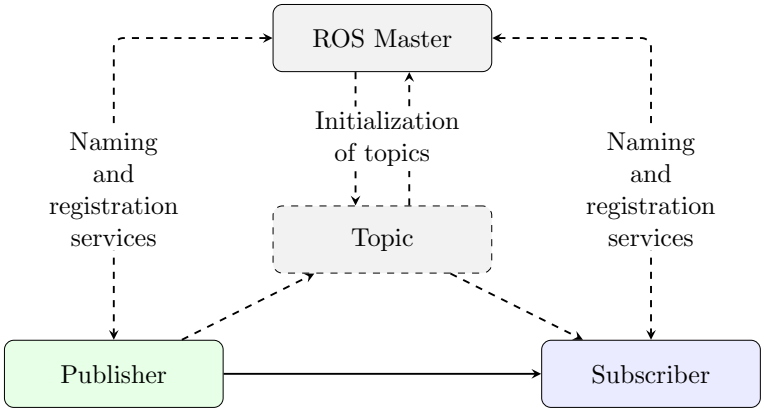

Figure 11: Publisher-Subscriber pattern structure layout in the ROS network. The master only initializes the topic, after initialization is done, communication is don peer-to-peer between Publisher and Subscriber. The Master also handles initialization of the nodes them selves. This means that dashed lines are valid during initialization phase, after that the full line is the only active connection.

5.3.1 ROS Master

The ROS master is an organizing unit responsible for naming and assigning addresses between nodes. When connection is established between ROS nodes, the information transmission is handled peer to peer. The version "Lunar" of the ROS master is set up on the user PC running Ubuntu 16.04.

5.3.2 MATLAB in ROS

MATLAB offers the Robotics System Toolbox which enables a MATLAB-instance to set up a ROS Master connect to an existing ROS network, though only over the slower protocol alternative TCP. Since the introduced delay is substantially lower the other times in the system this should not introduce any major problems. The publishing and subscribing in ROS is performed though custom commands in the toolbox.

5.3.3 Microcontrollers in ROS

Microcontrollers can virtually be set up as nodes in a ROS network using the package *ROS serial* which establishes an interface between serial communication and the ROS environment. Through this package an Arduino can request to subscribe or publish to a topic.

5.3.4 Visualization in ROS

The ROS suite is a Qt-based framework for GUI development called Rqt. This framework is adapted in Sjoerd Koopman's work [16] to make a GUI for setting up the connection to the Vicon PC and to visualize the streamed position from the proprietary Vicon Tracker.

5.4 Arm Controller and Trajectory Planning

This section covers the control implementation for the arm. Control of the arm is divided into two main part, the low level controller and the high level controller. The first being responsible for moving the actual arm and applies directly to each actuator. The high level controller is the supervisory controller and handles two main tasks, the first one being ball trajectory estimation and the second one being the LQR/trajectory planing for the end effector.

5.4.1 Supervisory Controller

After the ball has been tossed the ball trajectory estimator part of the supervisory control starts to sample the ball trajectory using the VICON camera system. After sampling a predefined number of sample points it estimates at what position the ball path will cross through the working space of the arm. This is done assuming that the ball movement can be considered as a projectile motion. This will be true if air resistance is neglected and the ball modelled as a point mass. It has been proven in [23] that the air resistance does not impact the results in a considerable way for a ball of this size. When the ball trajectory is established it is also possible to derive the speed of the ball during its trajectory and therefore also at what time the ball catching needs to take place. This produces the desired EE position in order to catch the ball, this position is then passed on to the high level controller.

The control part of the supervisory control is responsible for calculating how to move the arm in order to reach the desired EE position. This is done using

inverse kinematics. With knowledge of the dynamics of the arm and desired EE position as input it calculates how all three joints (shoulder, upper arm and elbow) should be actuated. The controller is implemented as a optimization problem using an Linear Quadratic Regulator (LQR) controller. Stating the problem of moving the arm as a minimization problem allows for the use of weight matrices that punishes different parameters. In this case the parameters being punished are absolute error between the current and desired position of the EE, the amount of usage for each joint and also how close to the joint limit the joint is used. The last part is implemented in order to avoid exceeding the physical joint constraints. To be able to implement the LQR and solve the inverse kinematics it has to be solved numerically, there are several methods for doing this and a few of the most common ones used are Newton–Raphson (NR), Gauss-Newton (GN) [24], steepest/gradient decent (GD) and variable metric (VM) [25].

One method that has been proven to converge fast and more often finds a solution is the Levenberg–Marquardt (LM) method [25]. In [26] a slightly altered version for the LM algorithm is proposed and proven to solve some of the problems faced when using LM for inverse kinematics. This is also the method used in this project, it has been further altered in order to take joint constraints into account. The iterative solution to this is implemented as following.

$$q_{n+1} = q_n + H_n^{-1} g_n \tag{1}$$

$$H_n = J_n^T W_e J_n + W_n + ((q_{lim} - q_n) W_l (q_{lim} - q_n)^{-1} \tag{2}$$

$$g_n = J_n^T W_e e_n h \tag{3}$$

Here q represents the vector containing the three joint angles, J is the jacobian matrix of the system and W_e, W_n are weight matrices for error and damping/usage of separate joints. q_{lim} is the joint constraints and is quadratically multiplied with the weight matrix W_l. In Equation (3) e it the error and h the step size for the error iterations.

The high level controller outputs three trajectories containing angle references $q_0 ... q_N$ where q_N represents the angles that gives the desired EE position. When followed, the trajectories ensures that the EE moves to the desired position in such an efficient way as possible considering the weight matrices. These trajectories are passed down to the low level controllers that are responsible to actually preform this movement.

5.4.2 Low Level Controller

The arm is of a serial nature with three DOF, resulting in three independent motors to be controlled. Each electric motor is connected to a microprocessor in the form of Arduino Unos. The low level controllers are classical PID regulators which will handle reference tracking of the planned trajectories calculated by the supervisory controller.

When the optimal trajectory for the EE has been planned by the high level controller, an array of the trajectory are sent via ROS to the Unos. The Unos receives the encoder values from the Arduino Nanos via serial communication. The PIDs uses the received trajectory as references with the encoder values as feedback to ensure closed loop reference tracking. The PID controllers are implemented on the Unos which calculates a control signal on the form of a duty cycle varying between 0-100% for a PWM signal with fixed frequency. The PWM signal is sent to the drivers powering the motors as well as a direction signal controlling the direction of the rotation.

The PIDs are tuned manually utilizing the completed system and using step responses until a satisfactory result is achieved. The goal is to have a quick, responsive and precise system, this means driving the steady state error close to zero while maintaining little oscillation and overshoot. Since the end position of the EE is crucial to the catching the PIDs are mainly focused on precise movement rather than speed

During operation the position of the various inertia's will shift due to the movement of the arm. This will lead to irregular forces that depend on the position of the arm which greatly influences the PIDs capability to successfully control the system. This leads to a desire to be able to change the control parameters during operation which is achieved using a form of state machine which implements different control parameters depending of the angles of the three joints of the arm.

The states will be divided into different areas of the work-space for the arm. Where these areas are defined by the various angles for each joint. Each state will have tuned control parameters giving a good system response for the corresponding arm configuration. Hence, depending on what state (joint angles) are currently active the control will have the corresponding control parameter. This will in some sense make the PIDs dynamic since they will be able to change control parameters during operation. The important part is to divide the work-space into states and tune the controllers to achieve satisfactory result in each given state.

6 Verification and Validation

Both verification and validation are essential procedures for checking that the system meets its requirements and specifications. In order to accomplish this, the individual subsystems were tested individually as well as the whole system after complete assembly.

6.1 Base Stability

It is critical to verify that the whole system will have the necessary stability even with the presence of vibrations. The base is deemed the most important part of the mechanical system to comply with this as the entire arm is attached to it. To test this, small and frequent impulse forces were applied at the top of the base. By doing so, the system was tested to observe its damping component which was deemed sufficient upon observing how little it vibrated and how quickly the vibrations died out as a result.

6.2 Actuation

The motors must be tested upon the complete construction of the arm to verify that enough torque can be provided by each motor to perform its action successfully. This was done by monitoring the reference and actual angles for the motor, and subsequently verifying that the motor is capable of reaching the reference angle. The process was repeated by setting the reference to cover the range of motion expected of the arm during juggling.

6.3 Supervisory Controller

As mentioned in the implementation section there are two main parts of the supervisory control: trajectory planning and inverse kinematics.

6.3.1 Verification of Ball Trajectory Planning

The trajectory planning must be verified to check that the estimated coordinates for the landing of the ball are sufficiently accurate. This is done after the ball trajectory calculations are made, and continued sampling of the ball location is taken to observe where the ball has actually landed. The tests were performed repeatedly by throwing the ball vertically in the air by hand.

6.3.2 Verification of Inverse Kinematics

The inverse kinematics was also tested to verify that a realistic and satisfactory arm motion was produced. This was done by holding the ball at several angles that can be physically reached by the arm and observing the coordinates. Those coordinates were then input to the code of the inverse kinematics alone and checked to see that the motion produced was the desired one.

6.3.3 Verification of Elapsed Time

An imperative part relating to the entire Supervisory Controller is timing. It is essential that the entire code runs sufficiently quickly so that enough time is given for the actuation. Given a vertical toss of 2.5 m (defined by requirements in chapter 1.3), the total air time for the ball is about 1.5 seconds. The stopwatch timer was used throughout the supervisory controller code in MATLAB to monitor the execution times of individual parts of the code as well as the total time required to have all reference angles ready to be sent over to the Arduino Unos. It was estimated that it would require about 1 second for actuation of all three motors to finish. Therefore, the elapsed time must be verified to be under 0.5 seconds for an ideal toss, in order for the system to successfully work.

6.4 Tossing Motion

The tossing motion is verified through the motion of the arm and height of the throw.

6.4.1 Verification of Tossing Motion

The tossing motion must be verified to see that tossed balls land at positions which are reachable. To test this, a toss is performed and using the trajectory planner and LQR software, it can be verified that the LQR has not reached "break condition": meaning that that it is performing more than a very big upper limit of iterations to reach the desired position.

6.4.2 Validation of Tossing Height

In accordance with Section 1.3, the robot should be able toss the ball 2.5 meters above the base. This was tested with the trajectory planner, where the motion of the ball can be tracked and then plotted on a graph.

7 Results

This section provides the results of the verification and validation tests that were made for the subsystems as well as the system as a whole.

7.1 Base Stability

Initially, the base was very prone to vibrations. Impulse forces applied to it would produce vibrations that took a long time to decay, indicating that the damping was not sufficient. This was mainly due to the construction. The base was fixed only from its bottom end to the ground while the top was free, giving it a cantilever-like structure. A solution was proposed; add an extra fixture to the top of the base so that the base is fixed from both of its ends. The result was encouraging as the response from the impulse force died out much quicker due to this design change.

7.2 Actuation

All three motors exhibited normal behaviour where the PWM signal required to move the arm was not unreasonably high, indicating that the motors are capable of providing the required torque for performing the juggling motions without having to draw too much power or push the motor to its physical limits. But the Upper-Arm Motor has a tendency to become hot if run continuously for longer than five minutes.

7.3 Supervisory Controller

The overall supervisory controller was deemed satisfactory. Both ball estimation and trajectory planning worked as intended.

7.3.1 Accuracy of Trajectory Planning

The error was calculated to be below 10 cm off from the expected point as the worst case and about 4 cm off on average. This was deemed sufficient as the size of the EE is wide enough and therefore, it would be able to catch the ball under such conditions.

7.3.2 Motions of Inverse Kinematics

The inverse kinematics solver, using the Levenberg-Marquardt algorithm, was able to produce motions that the low level controllers could implement. At some times the optimal solution was one that made the EE be positioned in such a way that the it would not be able to be kept leveled. This was solved by adding non-symmetric constraints in the solver and re-tuning the solver.

In Figure 12 the planned movement can be seen when the initial position is [0.11, 0.34, -0.94] m and the final desired position is [0.22 0.77 -0.38] m, where the origin is set in the shoulder joint. Movements like this was generated then implemented and used to catch the ball, by test like this it was verified that the generated path and the actual path where close enough to each other to be considered the same.

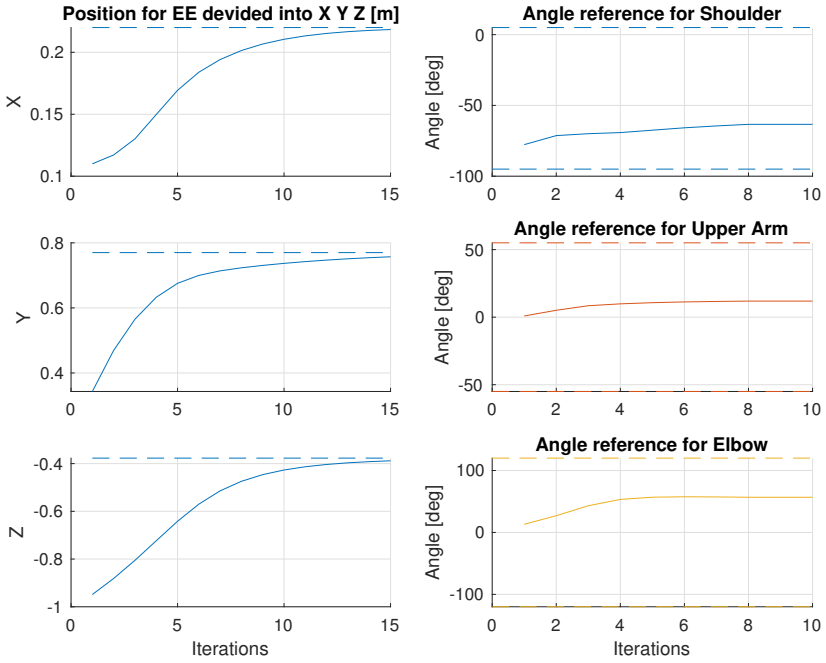

Figure 12: Left side shows the world frame travel trajectory for the EE, dashed lines is the position goal. Right side shows the angle references generated by the LQR that is passed on to the LLC, dashed lines show the joint constraints.

7.3.3 Elapsed Time

Following code optimization and tweaks, the execution time from start to finish was estimated to be about 0.6 seconds. Graphical presentations like plots, prints and unnecessary calculation(e.g percentage of trajectory error) were excluded. This was too long and it proved to be unsuccessful on most throws as the height of the tossing motion was lower than expected.

In order for the LQR to perform properly, it requires the initial angles for the motors. Initially, those angles were read from the Arduino Unos via ROS after the tossing motion had been performed. However, this proved to be costly as the time it took to read all three encoder values were cumulatively about 0.3 seconds, which is 50 percent of the execution time. Hence, those lines were removed and instead the knowledge of the tossing motion were utilized to determine those angles. Having the tossing motions hard-coded in the Arduino Unos, the angle for each motor could be determined.

This proved fruitful as the arm reached the end position, by the time the ball had reached the maximum height.

7.4 Tossing Motion

The results of the validation and verification of the tossing motion are presented below. Note that the tossing motion did not reach the height set by the technical requirement of 2.5 m.

7.4.1 Verification of Tossing Motion

The tossing motion varied greatly due to three main factors: 1. The EE and its angle around the x and y axis. 2. The reference angle set for elbow motor to reach -with 100 percent PWM- to throw the ball 3. The reference angle set after the elbow motor has reached this angle, to decelerate the elbow and perform the toss.

The servo motors controlling the EE along with its IMU were unreliable at best and that created problems with keeping the EE horizontal throughout the tossing motion. Therefore, it was decided to fix the EE so that it would remain at a 90 degree angle with the lower arm. Upon doing this, the primary source of throw accuracy was now to be the two reference angles for the throwing motions. They were tweaked continuously until the ball was landing consistently at positions that the arm can physically reach. This was confirmed by the absence of "Break condition" when running the algorithm.

7.4.2 Validation of Tossing Height

The tossing height was calculated to be about 1 m, above the base. This was below the requirement of 2.5 m. The reason for this is mainly due to the elbow motor performing the tossing motion, without the help of the shoulder. It was performing the motion at full speed and power. The conclusion that can be drawn from this is the following:

1. For the lower arm to perform the tossing motion alone, it is required to have a motor that is capable of higher rotational speed.
2. For the present design to perform the tossing at the desired height, the shoulder must contribute, which cannot be done by simply moving both the shoulder and elbow to a set reference, at the same time. This is because the elbow motor is faster than the Shoulder Motor which will result in a sequential movement that is not desired. Therefore, a complex motion will have to be performed using kinematics instead.

8 Discussion and Conclusion

In this section different subsections of the project are discussed and finally concluded.

8.1 Hardware Implementation

The overall functionality of the arm was satisfactory. The mounts for all the individual actuators were designed carefully and everything fit well together. By using steel and aluminium close to the reference frame and 3D-printed parts out towards the EE the arm maintains some rigidness without adding to much weight to itself. This is essential in order for the arm to be both fast and precise.

The elbow motor functioned very well. It was capable of making fast accelerating motion required for tossing. The large torque of the motor also made it easy to control.

The play in the gearbox of the Shoulder Motor proved to be a problem quite early. It had a poor effect on the EE positioning accuracy and the smoothness of the motion. The play could be explained by the old age of the motor and it caused the internal components of the gearbox to be exposed to large forces, causing it to break down a number of times. By ordering a brand new motor with gearbox this could have been avoided.

Due to a lack of gearbox the Upper Arm Motor was really fast and weak, making it quite difficult hard to control and keep at one position. This was solved by applying friction to the outgoing shaft by putting a 3D printed dampener on the shaft, slowing down the shaft but also helping to dampen the effect the asymmetric moment of inertia around the shaft had on its controllability. Another fix for the problem would have been move this actuator movement to the shoulder. This would also reduce the bad effects on of the non-symmetric load that exists due to the mounting of the elbow motor.

8.2 System Architecture

The communication between all nodes works really well when set up. The serial ROS communication between the Ubuntu PC and the Arduinos is set to a baud-rate of 2 000 000 without any problems.

The system has a large amount of nodes for communication. Although the communication is fast enough, booting up the system is complicated and time consuming. The nodes have to start in a specific order to boot, with

the Vicon system and the ROS master node being started simultaneously on the Vicon and Ubuntu User PC respectively. Vicon has to be set up on the Ubuntu PC with the correct IP address and set to track markers. Following that, the Arduino nodes have to and the MATLAB node have to be started up in separate terminal windows.

After the aforementioned nodes have been set up the, the actuators of the robot needs to be connected to their power supply. Note that leaving the power on between sessions is not recommended due to safety concerns, and during code upload the reference values are undefined. Uploading new code to the low level controllers also meant that the nodes need to be shut down, otherwise the lost sync error will be flagged by ROS. Also note that if multiple Arduinos are plugged in when uploading code, there is a risk that the code will upload to the wrong board. Therefore the other Arduinos have to be unplugged.

The high communication speeds gave minimal time delays. The data packages sent through serial were compressed into Int16 to match the resolution of the hardware. The for the system architecture were satisfactory, further improvements are discussed in Section 9.

The large amount of nodes are simplified by working in parallel. Planning the system architecture from the beginning of the project minimized communication issues between the nodes.

8.3 Supervisory Control

Using ROS was a good choice in retrospect, allowing for quick power up and making the system more modular. The performance was limited however in the current set up since the recording of points necessary to do ball trajectory estimation, and the HLC were running within the same process. This design provides a convenient user interface but makes it impossible to simultaneously record ball trajectory data points for the estimation and do the HLC motion planning. Instead, an initial set of points are recorded (if the apex is 1.5 m then this corresponds to a quarter of the trajectory) and the full ball trajectory extrapolated from that. The HLC motion planner works on that and hopefully the ball does not deviate. A superior way however would have been to separate these two into two processes, such that while the HLC motion planner is doing its calculations based on a first ball trajectory estimate, the other process can continue working in the background, providing online ball trajectory estimation.

With a tuned focus of the cameras and a low number of moving bright objects in the room, the captured data is really precise. This makes an interpolation sufficient to extract movement in the horizontal plane. To

further increase the robustness of the system a filtering technique could have been implemented together with a higher sampling rate of the cameras. Though, increasing the sampling rate would also increase the computational cost which could have been compensated with higher performing computer.

The lack of time to tune frame rate and the number of samples recorded before making a predicted catching position could have influenced the performance of this algorithm. This was due to the inability to start full-scale testing before the mechanical arm was in place. To avoid this issue a possibility would have been to create a software model to create a SIL environment for the controller, something that unfortunately was not completed due to being down-prioritized.

8.4 Low Level Control

Overall the LLC was sufficient to carry out the deed of position the arm to a requested location. It could receive instructions from the ROS network quite well, the encoding and the communication between the encoder and controller ran without any perceived noise or missed encoder steps, and the controller could produce actuator control signals for the driver without fail. With that being said, there were two issues that were encountered. The first is a fundamental problem with the entire LLC approach, and the second a hardware limitation.

8.4.1 Static PID Challenges

One of the inherent issues associated with using serial mechanisms is that each actuator experiences an additional DOF worth of varying moment of inertia, such that - ignoring the effect of the EE - the final actuator has a fixed moment of inertia. This had bad implications on the control of the system's efficiency, speed, reliability, etc; as the PIDs were manually tuned to work optimally in a certain angle range and for a specific pose of the robot. It means that a 10 degree angle change moving from 0 degrees to 10 degrees, and moving from 80 degrees to 90 degrees, could require an order of magnitude difference in supplied power.

As mentioned earlier in Section 5.4.2 an ad-hoc solution using differently tuned PIDs for different working ranges was attempted. It worked quite well for the Shoulder Motor, which has a worm gear with a big gear ratio. The self-locking aspect of it meant that the Shoulder Motor was much trivial to keep the arm at an angle once it was reached, but the aforementioned challenges with the changing loads nonetheless meant that the speed

with which it rotated 10 degrees was very different for different working ranges.

Having this kind of multiple manually tuned PID is not feasible in the long-term. Any kind of future change can require the PID to be manually re-tuned, and that workload is of course multiplied by the amount of angle range divisions that have been made.

8.4.2 Arduino Controller Limitations

Originally the team worked on getting a BeagleBone Black Single-Board-Computer with two integrated Programmable Real-Time Units (PRUs i.e., microcontrollers) to do the controlling of the 1-DOF model, but this was abandoned due to lack of sufficient documentation. Instead the group turned towards Arduino, especially for he the controller. This allowed the group to be able to re-use code from previous courses, as well has have an extensive user-base to fall back on for troubleshooting.

The Arduino Nano does a good job as an encoder, its small size making it particularly convenient. The UNO on the other hand is limited. The first problem is that it only has one hardware serial port. That means that the encoder has to send its outputs to a software serial, i.e. an emulated serial port using software. In this case a timr is used up to create one serial port, Another timer is taken up ROS for serial communication with the ROS network. Finally, one timer is necessary to create a PWM pin for the driver, occupying all three timers. If one more timer had been available, the controller loop code could have been run with timer to get a constant sampling frequency, as opposed to simply running as fast as possible with varying sampling frequency. Having the sampling frequency vary adds another source of uncertainty to the whole control process.

8.5 Conclusion

The project accumulated in a satisfactory result given the circumstances. The robotic arm was able to perform throwing and catching to some extent. The mechanical structure was well defined for the specified task and able to move in a rather large work-space through the serial mechanism. Some of the accuracy was lost due to the non-symmetric design where the inertia from the elbow motors was not fully centered around the axis of the upper arm making it difficult to control. This could be fixed by altering the design of the serial mechanisms with improvements to the motor attachments or counter-weights. However, this project serves as a proof that a serial-manipulator is

a good design choice for this task, since it was able to complete the desired movements while also keeping a human-like appearance to the robot.

The HLC design was fully functional and able to plan an optimal trajectory for the arm movement and send the reference trajectories to the low level PID controllers without too much delay. As discussed in Section 8.4.1 static PID controllers are very difficult to tune properly in a highly dynamic system due to the changes in inertia during movement of the arm. One way to improve the results would be to utilize a MPC controller for the entire system. Without the use of HLC and LLC the MPC would plan the trajectory for the arm and actuate the motors. This would lead to a optimal adaptive controller able to handle the varying inertia's. The MPC controllers would need a accurate representative model of the robotic arm which could be built now when the arm design and motors are defined. This could lead to an increased accuracy and capability of the robot while also decrease the need for communication.

The Vicon camera system used was highly functional and able to track the movement of the ball. However, the placement of the cameras could be further reviewed to find an optimal configuration. Moving the cameras closer to the objects it could hopefully increase the accuracy of the measurements and placing the cameras symmetrically around the operating point would also further increase the camera systems capability of tracking the object. A major issue occured during the demonstration of the robot, namely that the cameras views were blocked by obstacles which made it impossible for the cameras to track the ball below a certain height.

The EE need to be reworked with stronger servo motors. During the catching motion, the EE was not able to counter-act the impact from the ball when landing in the cup and thus unable to catch it properly. Another issue is the vibrations generated in the EE during the catching which causes too much measurement noise in the IMU. Therefore, the IMU should be placed within some form of damping material. Another good solution to the EE would be to have a rotational motor in the forearm of the robot, leading to only have a tilting motion at the EE thereby increasing the stiffness of the design.

8.5.1 Project Procurement Limitations

One non-technical issue that limited the project was that a lot of the motors that were found could not be ordered. Many of the bigger vendors found require company accounts be registered, and as this project was done under KTH that was not an option. Furthermore, KTH requires all orders be done using invoicing. These two factors left the group with a much narrower list

of vendors and products, many of which were out of stock and required long lead times.

Fortunately, as a KTH project the group also had access to motors from old KTH projects. The Elbow Motor, which was thought to be especially critical was picked out and ordered using requirements gained from calculations and simulations, but two other motors (the Upper-Arm Motor and Shoulder Motor) were found in KTH storage.

8.5.2 Ethics and Sustainability

The main ethical issue encountered in this project was the use of carbon fibre. As mentioned in [insert reference to part where we compare carbon fibre, steel, and aluminum], by choosing the forearm part connecting the elbow motor and the EE to be made in carbon fibre, we had a part that was not only incredibly light but also incredibly stiff and strong in the axial length. As this part was farther away from the first actuator as well as quite long it was important that it was very light as the torque necessary to actuate it might otherwise have been prohibitively large. Carbon fibre does however come with some problems that need to be taken into consideration.

The first problem is the problem of its fibres. The carbon fibre tubes we ordered, called Excel Ultralite, were bought from a reseller of a Finnish company and arrived as two 1500 mm 35/31 mm diameter (outer and inner respectively) long tubes. Only one of the tubes ended up being used, and it needed to be cut to about a third of its length. Cutting it however was an arduous process, requiring one person to saw through it with a handsaw made to saw through metal. The process took around half an hour, and while one person was sawing another helped to clamp and unclamp it to rotate it. A third person made sure to keep a vacuum cleaner nozzle close by the saw to clear out all the fibres. Afterwards, the room and its table tops were vacuumed, as well as the pieces themselves.

Despite this pre-caution however, and especially afterwards when handling the two halves, members reported micro-lacerations and heavy itching as the microscopic fibres migrated towards their hands. They were forced to repeatedly wash their hands with soap under near-scalding hot water, until the tubes were sealed with liberal use of duct tape. Furthermore, though this might not have been experienced by the group members, the coating that is used on the carbon fibres can cause chemical irritation. Possible health effects of this include irritation of eyes and lungs, dizziness, drowsiness, nausea and even vomiting.

A second problem is that of its environmental impact. Carbon fibre, due to its light weight, can cause a significant reduction in fuel costs when used

in automotive and aerospace applications. While steel and aluminum can be relatively conveniently melted down and recycled, products using carbon fibre are more difficult to recover value from, hindering their use in a circular economy.

Thus, from an environmental and social sustainability perspective, the use of carbon fibre is not completely straightforward, and there are definitely ethical concerns to be taken into consideration. For this small scale project it did not make much of an impact.

9 Future Work

In this section possible improvements to design, structure and implementation are discussed.

9.1 Basic Motor Replacements

The Shoulder and Upper-Arm Motors were found on KTH and should to be replaced as soon as possible. The Shoulder Motor is a motor equipped with a worm gear. This worm gear unfortunately contained play until it eventually broke down. The worm gear internally contains screws connecting through shafts to transfer torque between them. The worm gear's screws gave out completely the week before the presentation day. It was repaired, broke down again, repaired and finally broke down just in time for the demo. It was tied down to the frame and the shoulder was not used.

The Upper-Arm Motor is functional but would see significant improvement with a gear box attachment. At its current state, without any gear box, it is fast but weak. When put under load it slows down and compensates by drawing high amounts of currents, which subsequently causes it to overheat. 3D printed PLA attached to the shaft holding the encoder reader in place actually warped during testing. An argument can be made that a motor significantly smaller but also significantly geared up with an appropriate gear box could be more effective than the current setup.

9.2 Potential Mechanical Improvements

In the first timeline set in May, the plan was to have had the mechanical arm designed and the parts for it ordered before the summer, so that the group could arrive in September. Unfortunately the work necessary to create a sophisticated model capable of providing realistic baseline velocity and torque data, which could be used as baseline data by which motors could be chosen and ordered, required a lot more work than expected. In the end, one intern at the department spent a whole summer working full time on creating that model for us. Furthermore, as mentioned in Section [insert ref to project procurement limitations], there were other external project limitations that further exacerbated the challenge of designing a mechanical arm.

In the end, with the help of the laboratory, workshop and prototyping assistants; an arm was made. But to anyone looking to further develop this project, the mechanical arm itself can definitely be looked into. Here are some starting points.

9.2.1 Placement of Upper-Arm Rotation Actuator

The robot arm developed in this project features a rotational motor in the upper arm, allowing the EE to move laterally in the transverse plane. In a human body this holds true as well. But the human shoulder also allows for a rotational movement in the transverse plane, instead of just the sagittal plane. One possible future work would be to investigate the benefits of removing the Upper Arm Motor in the arm and replacing it with a motor before the Shoulder Motor capable of rotating the EE in the horizontal plane, and the whole robot in its axial plane. This potential alternative actuator would be facing up and have the Shoulder Motor attached perpendicular to it.

The advantage that this would bring would be to shift weight from the arm and concentrate more of the weight to a single point on the frame. The frame is very sturdy and a "2 Shoulder Motors, 1 elbow motor" configuration might be easier to control. This could open up for stronger motors with big gear boxes.

9.3 End-Effector

The EE developed in this project did not have the resources necessary to develop it well allocated to it, and in the end it ended up being a big source of problems. Effort should be put into doing the following:

- **Fix Drift:** It was noticed that the EE's definition of level drifts over time. This could be a result of a number of influences, including vibrations, the force of the impact, sensor problems, inefficient or inadequate code, lack of sensor fusion, and so on.
- **Fix the Design:** As it is now the EE cup cannot keep level in all positions, and is liable to turn into and break itself. It needs to be re-designes so that it can rotate at least 90 degrees in every direction.
- **Increase Speed:** The EE is very slow compared to the rest of the movement of the forearm it is attached to. This means that it cannot adjust its position to be level fast enough.
- **Improve Strength:** The EE is very weak, making it too weak to work against the impact of the ball landing in it. Multiple times while testing the EE would fail to catch a ball because the ball would hit it and it would tip over.

The second improvement does not require much design work at all, while the last two require a new set of servo motors. The first one however could require quite a bit of work to troubleshoot.

In-hand manipulation was not a stakeholder requirement for this project, but it should be considered when moving forward. For example, a simple latch that can hold a ball in place and then precisely retract itself at the right time would be very useful.

9.4 Improving the Low Level Controller

The low level control experienced some fundamental issues. As mentioned in Section 8.4, the static PID controllers used are particularly limited in this type of multiple DOF serial mechanism, as the moment of inertia changes rapidly for all but one actuator.

9.4.1 Feedforward Control

One way to tackle this problem would be to add feedforward control to produce a baseline control signal, and then complement it with a signal from a feedback controller. The plant would need to be modelled, and the efficacy of the feedforward controller would be completely dependent on the quality of the model. The lower the model accuracy the more the feedback controller would need to be relied upon.

Here is a workflow for how a complete feedforward and feedback system could work:

1. Aggregate the different encoder values in order to make a digital reconstruction of the current pose of the robot.
2. Use the dynamics model of the robot to calculate the downward forces experienced by the motors due to gravity. These need to be counteracted by the feedforward control.
3. Use motor models, using known motor parameters, to calculate control signals in voltage.
4. Send the calculated feedforward control signals to each respective actuator.
5. Each actuator uses the feedforward control signal and corrects for any error with feedback control.

This requires a fundamental architecture change: the addition of a Master Controller of some kind. The Master Controller, containing a model of the complete system, would be receiving the individual encoders' angle information and calculate feedforward control signals from that information. There are a lot of possible ways of implementing it, and the following questions should be answered:

1. Can the Master Controller be run as its own ROS node on the computer, or should it be run on new embedded system hardware?
2. If it is new hardware, should the existing hardware be left as it is while the Master Controller figuratively "listens in" on the encoder and sends feedforward control signal "suggestions" to the controllers; or should the controllers be completely subsumed by the Master Controller and have it do the calculations and produce the PWM signals for the drivers?
3. If the latter is the case, would it possible to also perform the work of the encoders without any risk of missing counts?

The further down in the question tree one goes, the greater the demands placed on the Master Controller. The CPU would need to be very fast, orders of magnitutde over the 16 MHz of the Atmega328p found on the Arduino Unos and Nanos.

As mentioned in Section 8.4.2 the Arduino Unos used as controllers have in some aspects already reached their limits. Regardless of whether a Master Controller is added or not, they should at the very least be upgraded to a platform with more timers and hardware Serial UART capabilities. As of the writing of this report, superior microcontrollers in the same price range of the Arduinos include but are not limited to the Teensy boards, the Nucleo Mbed boards, and Cypress PSoC boards.

9.4.2 Current Control

In this project, an LQR controller was used that calculated the optimal trajectory for the arm. This LQR controller produced a list of positions, and thus, position control was used in the Low Level Controller. An argument could, however, be made for controlling the current instead.

At the moment, the actuators work to follow the positional references as they come, without any idea of how the actuators should pose themselves with respect to time. The idea is that by simply making sure that the positional references come in fast enough and that the power supplied is at a sufficiently high enough voltage to create the velocity necessary, the robot will be capable of throwing the ball high enough to go way beyond the requirements set by the stakeholder. Note that the position is actually not controlled directly, but rather by setting the velocity and then measuring the position.

Instead of controlling the position, however, it is be possible to control the current. Current is proportional to torque, and torque is what is necessary to accelerate the EE and the ball. The positional control has the advantage

that it is safer as it is possible to ensure that the robot arm does not hit itself or its frame by exceeding joint limits, as well as possible to ensure that the ball leaves the EE from a specific angle. But controlling the exact acceleration with which the ball leaves is not impossible.

If a simple positional control system keeping track of the joint positions was introduced in parallel with the current control system, it might be perform better and be able to do a more controlled throw. This becomes increasingly important as the number of juggling balls increases, which is ultimately the final goal.

9.5 Improving the High Level Controller

The High Level Controller implemented in this project relies on a soft-constrained LQR. The LQR does a adequate job producing positional joint angle references for the controllers to follow. The controller is quite sophisticated by itself. However one major flaw with the current design is that the LQR lacks any mechanical system knowledge. It also possesses limited knowledge about the dynamics of the system, as only lengths and angles were included. This is due to the fact that no accurate enough model for the dynamics and mechanics of the system has been derived. The limited scope of this project did not allow for detailed modelling since the control system had to be developed in parallel with mechanical design and ordering parts, this made it hard to model something that was non-existing. Another factor that lead to the suboptimal design of the LQR was the lack of knowledge about many of the motor parameters. If known it would have been possible to take motor torque and speed into account when doing the LQR calculations. This is something that can be improved and deriving a detailed model for the system would have made it possible to both improve the LQR but also to move over to a more sophisticated controller such as model predictive controller (MPC).

Since the structure of an LQR does not allow for hard constraints there was no guarantee that the joint limits would not be violated. This could be solved by implementing an MPC. With the MPC and a detailed model of the system the need for the LLC is removed. This means that the supervisory controller can actuate the motors directly and this removes time delays. With a detailed system model the problems faced with the dynamic PID tuning are also mitigated.

9.6 Improving System Architecture

For designing and developing reasons the system architecture is a good choice. The decentralized node structure makes it simple to edit individual nodes and add more nodes. When the design is finalized, an optimization can be made by compressing the system to a hardware like Beagle Bone Black. The Beagle Bone Black will run both the high and low level controllers. The Beagle Bone Black has two independent PRUs which can handle the real time low-level control. The man processor is connected to the PRUs with high speed busses. This will eliminate some of the computing power and time delays from the communication.

9.7 Gazebo

When the project was originally pitched, a software known as Gazebo was brought up as a tool for simulating the robot continuously while at the same time building it. ROS is used in the project to facilitate the information flow between the various nodes in the network, whether it be the low level Arduino Unos controlling the individual motors, to the Vicon camera system, or the MATLAB regulator software. The benefit of using Gazebo is that it fits seamlessly with the ROS software. Within Gazebo, a digital version of the robot can be made. The Arduino Unos could be fed simulated movement data from this virtual robot attempting to juggle a virtual ball, and their responses logged and used. This allows for potentially powerful X-in-the-loop type testing and development.

At some point early in this semester however it was decided to let go of Gazebo. Initially the group relied on simulations done in MATLAB/Simulink, based on previous work done over the summer by an intern at the institution mentioned in Section 5.1.2. Those simulations provided enough data to begin the procurement process of the motors, but the intricacies of the Mechanical design were far from finalized. Thus, the group agreed that since some modelling had already been done, attempting to redo the work with Gazebo, a software no one in the group had encountered before, would be a distraction from the actual work. In order for Gazebo to really pay off and give us results beyond what we had already gotten, every other system would first have to be set up, debugged and finalized. Also, the mechanical design would have to be quite fully set, from the dimensions of the parts to the choice of material. Allocating a person to work on learning Gazebo would take away manpower needed to complete the aforementioned things that would make Gazebo effective in the first place.

Now that the surrounding systems have been set up, and a first prototype arm model has been made, it is definitely worth re-considering Gazebo. The

range of systems and complexities will only increase as the robot goes from catching and throwing one ball, to adding an arm and balls. Having Two arms working in conjunction to juggle balls between each other does not just double to complexity of the system but adds to it non-linearly, as not only does everything already done need to be repeated but the arms, which are essentially two separate robotic systems, need to be work in collaboration. Gazebo would definitely help in developing such a big system.

References

[1] L.-W. Tsai, in *Robot Analysis, The Mechanics of Serial and Parallel Manipulators*, vol. 1, 1999, 17–48 vol.1, ISBN: 978-0471325932.

[2] N.-N. D. D.-D. G. Niquín-Herrera, "Complete serial mechanism analysis: A screw theory approach for mobility study", 2017-09.

[3] P. Wenger, C. Gosselin, and B. Maillé, "A comparative study of serial and parallel mechanism topologies for machine tools", 1999.

[4] Y. Zhang and K.-I. Ting, "Design and analysis of a spatial 3-dof parallel manipulator with 2t1r-type", *International Journal of Advanced Robotoc Systems*, 2013-02.

[5] J. Wang and C. M. Gosseli, *Singularity loci of a special class of spherical 3-dof parallel mechanisms with prismatic actuators*, Accessed: 2018-05-11, 2004.

[6] I. A. Bonev, "Geometric analysis of parallel mechanisms", PhD thesis, de l'Universite Laval, 2002-11.

[7] X.-M. Niu, G.-Q. Gao, X.-J. Liu, and Z.-D. Bao, "Dynamics and control of a novel 3-dof parallel manipulator with actuation redundancy", *International Journal of Automation and Computing*, pp. 552–562, 2013-12.

[8] E. H. G. Yang W. Chen, "Design and kinematic analysis of a modular hybrid parallel-serial manipulator", *7th International Conference on Control, Automation, Robotics and Vision*, 2002.

[9] M. Rahman, S. K. Sarkar, S. K. Das, and Y. Miao, "A comparative study of lqr, lqg, and integral lqg controller for frequency control of interconnected smart grid", in *2017 3rd International Conference on Electrical Information and Communication Technology (EICT)*, 2017-12, pp. 1–6.

[10] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality", *Automatica*, vol. 36, no. 6, pp. 789–814, 2000, ISSN: 0005-1098. DOI: 10.1016/S0005-1098(99)00214-9.

[11] T. Oka, N. Komura, and A. Namiki, "Ball juggling robot system controlled by high-speed vision", in *2017 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, 2017-10, pp. 91–96. DOI: 10.1109/CBS.2017.8266074.

[12] T. Kizaki and A. Namiki, "Two ball juggling with high-speed hand-arm and high-speed vision system", in *2012 IEEE International Conference on Robotics and Automation*, 2012-05, pp. 1372–1377.

[13] A. Leosirikul, D. Chilin, J. Liu, J. F. Davis, and P. D. Christofides, "Monitoring of low-level pid control loops", in *2012 American Control Conference (ACC)*, 2012-06, pp. 5664–5669. DOI: 10.1109/ACC.2012.6314747.

[14] L. Wills, S. Kannan, B. Heck, G. Vachtsevanos, C. Restrepo, S. Sander, D. Schrage, and J. V. R. Prasad, "An open software infrastructure for reconfigurable control systems", in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, vol. 4, 2000, 2799–2803 vol.4. DOI: 10.1109/ACC.2000.878721.

[15] (). Vicon tracker [computer software]. Accessed: 2018-05-11, [Online]. Available: https://www.vicon.com/products/software/tracker.

[16] S. Koopmans. (2018). Vicon control. Accessed: 2018-11-12, [Online]. Available: https://github.com/SjoerdKoop/vicon_control.

[17] R. Rajkumar, M. Gagliardi, and L. Sha, "The real-time publisher/-subscriber inter-process communication model for distributed real-time systems: Design and implementation", in *Proceedings Real-Time Technology and Applications Symposium*, 1995-05, pp. 66–75. DOI: 10.1109/RTTAS.1995.516203.

[18] M. Montemerlo, N. Roy, and S. Thrun, "Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (carmen) toolkit", in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 3, 2003-10, 2436–2441 vol.3. DOI: 10.1109/IROS.2003.1249235.

[19] P. M. Newman, "Moos - mission orientated operating suite", Tech. Rep., 2008.

[20] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: An open-source robot operating system", in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[21] A. S. Huang, E. Olson, and D. C. Moore, "Lcm: Lightweight communications and marshalling", in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010-10, pp. 4057–4062. DOI: 10.1109/IROS.2010.5649358.

[22] B. E. Durmaz, B. Kaçmaz, İ. Mutlu, and M. T. Söylemez, "Implementation and comparison of lqr-mpc on active suspension system", in *2017 10th International Conference on Electrical and Electronics Engineering (ELECO)*, 2017-11, pp. 828–835.

[23] J. Kober, M. Glisson, and M. Mistry, "Playing catch and juggling with a humanoid robot", in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, 2012-11, pp. 875–881. DOI: 10.1109/HUMANOIDS.2012.6651623.

[24] H. Kushner and P. G. Dupuis, *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer-Verlag New York, 2001, vol. 24. DOI: 10.1007/978-1-4613-0007-6.

[25] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems",

in *Proceedings of the 2005, American Control Conference, 2005.*, 2005-06, 300–306 vol. 1. DOI: 10.1109/ACC.2005.1469949.

[26] T. Sugihara, "Solvability-unconcerned inverse kinematics by the levenberg–marquardt method", *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 984–991, 2011-10, ISSN: 1552-3098. DOI: 10.1109/TRO.2011.2148230.

Appendices

A Budget

All prices are given in Swedish krona and VAT is excluded. Borrowed components are excluded.

Product Name	**Unit Cost**	**QTY**	**Total Cost**
Combination padlock	69.90	3	209.70
Connector: FFC/FPC 6P - Molex	3.39	30	101.70
DC motor: DOGA DO31990593B00	2072	1	2072
CF tube: Hiilikuituputki 35/31	761	2	1522
IMU: AltIMU-9 v5 Gyro/Acc/Comp	179	1	179
Motor driver (h-bridge) 18v25A.	399	5	1995
Motor driver (h-bridge) 24v13A	429	1	429
Motor driver (h-bridge) 24v21A	409	1	409
Motor driver (h-bridge) 30v25A	399	2	798
Magnet ring: MR050C040B072B02	564	4	2256
Encoder: RLB2HDA05BF06C00	424	4	1696
Hub connection: Type 17 w flange	302	5	1510
Teensy 3.2	259	1	259
USB-B-Cable 5m	109	4	436
Prototype center service	300	5	1500
Total			**15372.40**

B Time Plan

A time plan was set twice during the course of this project. The first before the summer in May, and the second in October.

B.1 Time Plan Set in May

Inspired by the Agile philosophy and the Scrum method presented earlier in the course, the group decided to divide the then coming semester into a series of fourteen day sprints.

- **Sprint 1: Weeks 36-37: 03/09 - 16/09:** Assembly of parts for the arm ordered before the summer begins and completed. Final report 10% (structure is set up).
- **Sprint 2: Weeks 38-39: 17/09 - 30/09:** Electronics (Encoder and BeagleBone) are set up so that it can accept open loop inputs. Final report 20% done (Introduction and Background sections are filled in).
- **Sprint 3: Weeks 40-41: 01/10 - 14/10:** The whole system is put together with the Vicon camera system such that there is some form of feedback loop present. Assess if it is feasible for us to build a second arm and order the parts if so. Final report 30% done.
- **Sprint 4: Weeks 42-43: 15/10 - 28/10:** Exam Period. Final report 50% done.
- **Sprint 5: Weeks 44-45: 29/10 - 11/11:** The control systems is tuned and a simple one-arm juggle pattern can be done, i.e. throw a ball up and catch it. Final report 60% done.
- **Sprint 6: Weeks 46-47: 12/11 - 25/11:** Work on implementing a more complicated juggling pattern with the one arm. If there are two arms, have the second arm replicate the first one. Final report 70% done.
- **Sprint 7: Weeks 48-49: 26/11 - 09/12:** If possible implement a juggling pattern that requires one arm to throw and one arm to catch. Final report 80% done.
- **Sprint 8a: Week 50: 10/12 - 17/12:** The HK projects are shown off to the public so the robot must be as presentable as possible. Add clown hair and play circus music. Final report 95% done.
- **Sprint 8b: Week 50-51: 18/12 - 23/12:** Final report proofread and sent in.

B.2 Time Plan Set in October

While the Software and Higher Level Control parts of this project had progressed quite well, the Mechanical and subsequently Lower Level Control parts were behind schedule. Problems with procurement delayed and eventually made the eventual construction of a second arm not feasible. In this timeline weeks were used and not sprints.

- **Week 41** – Close the loop on the 1DoF and make an assessment of the latency, find carbon fibre suppliers and order them, do the kick off, basic LQR structure done,
- **Weeks 42 & 43 (EXAM WEEKS)** – Whole arm design and all the stresses checked for, find and order mechanical clamps, find gears for any extra motor, explore the machining and steel/aluminium supplies possibilities and availability here at KTH and possibly order those things.
- **Week 44** – LQR constraints done, motor parameters found and low level controller done, drivers ordered and fixed, machining done and finished, μ-controllers ready and progammed
- **Week 45** – integrate the controllers with the ROS. Assemble the arm and look into creating a stand. Connect the encoders with the μ-Controllers and drivers and batteries.
- **Week 46** – Arm done. Report begun.
- **Week 47** – Tuning different controllers, dealing with latencies and trying to optimise the code. Position verification and creating calibration methods.
- **Week 48** – Finalise the juggling motion.
- **Week 49** – Ready for presentation.
- **Week 50** – PRESENTATION

C Simulink Model

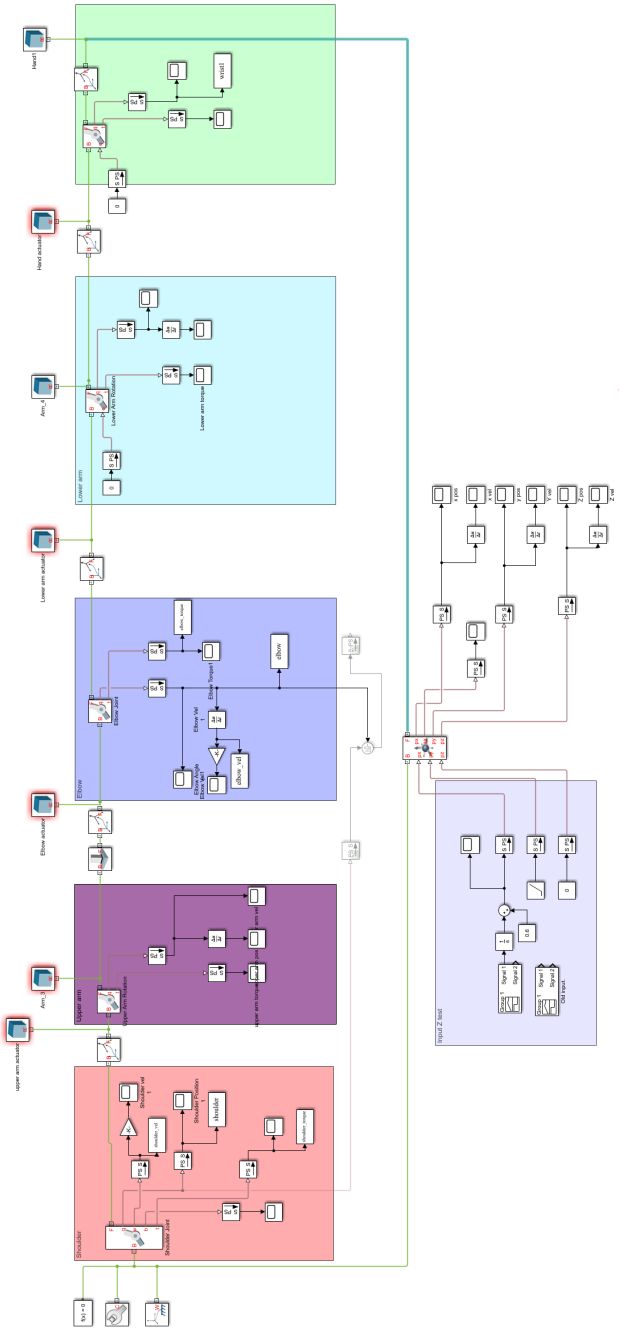

Figure 13: Simscape model of the mechanical system.

D Software

All software used in this project can be found in the JuRP-HK Github repository: `https://github.com/fschalling/JuRP_HK`